

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

MODUL 3

Prinsip-Prinsip Perancangan Class



Oleh:

Ramadhan Wijaya

2211102208

IF-10-M

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

Destructor

Destructor adalah metode khusus yang digunakan untuk membersihkan sumber daya yang dialokasikan oleh sebuah objek ketika objek tersebut tidak lagi diperlukan. Dalam bahasa pemrograman seperti C++, destructor dipanggil secara otomatis ketika objek dihapus atau keluar dari cakupan (scope). Ini memungkinkan penghapusan memori atau pembebasan sumber daya lain yang digunakan oleh objek. Namun, dalam Java, tidak ada konsep destructor yang eksplisit seperti dalam C++ karena Java menggunakan pengumpulan sampah (garbage collection) untuk mengelola alokasi memori dan pembebasan memori otomatis. Dengan demikian, tidak perlu mengimplementasikan destructor secara manual dalam Java.

Visibilitas Bagi Atribut dan Metode

Visibilitas mengatur tingkat aksesibilitas dari atribut dan metode dalam sebuah kelas. Visibilitas ini membantu dalam menerapkan prinsip enkapsulasi, di mana kita dapat mengendalikan akses ke data dan perilaku dari kelas-kelas yang berbeda. Ada tiga tingkat visibilitas yang umum digunakan dalam Java:

- **Public:** Atribut atau metode yang dinyatakan sebagai public dapat diakses dari mana saja, baik dari kelas yang sama, paket yang sama, atau bahkan paket yang berbeda. Ini memberikan tingkat akses yang paling luas.
- **Private:** Atribut atau metode yang dinyatakan sebagai private hanya dapat diakses dari dalam kelas itu sendiri. Mereka tidak dapat diakses dari kelas lain, bahkan jika kelas tersebut berada dalam paket yang sama.
- **Protected:** Atribut atau metode yang dinyatakan sebagai protected dapat diakses dari kelas yang sama, paket yang sama, atau kelas turunan (subclass) dari kelas tersebut, baik dalam paket yang sama maupun paket yang berbeda.

Accessor dan Mutator

Fungsi accessor (getter) dan mutator (setter) adalah metode yang digunakan untuk mengakses dan memodifikasi nilai dari atribut suatu objek.

- **Accessor (Getter):** Metode accessor digunakan untuk mengambil nilai dari sebuah atribut. Biasanya metode ini bersifat public dan hanya mengembalikan nilai dari atribut tertentu. Ini memungkinkan untuk membaca nilai atribut tanpa perlu mengizinkan akses langsung ke atribut tersebut dari luar kelas.
- **Mutator (Setter):** Metode mutator digunakan untuk mengatur nilai sebuah atribut. Biasanya metode ini juga bersifat public dan menerima parameter untuk mengubah nilai atribut. Ini memungkinkan kita untuk menerapkan validasi atau logika lain saat mengubah nilai atribut.

Penggunaan accessor dan mutator mematuhi prinsip enkapsulasi, yang menyembunyikan rincian implementasi internal sebuah kelas dan hanya mengekspos fungsionalitas yang relevan kepada pengguna kelas tersebut.

Method Overloading dan Operator Overloading

- **Method Overloading:** Method overloading adalah praktik memberi nama yang sama pada dua atau lebih metode dalam sebuah kelas, tetapi dengan tanda tangan yang berbeda. Tanda tangan metode mencakup tipe dan urutan parameter. Java mendukung method overloading, yang memungkinkan kita untuk memiliki metode dengan nama yang sama tetapi dengan parameter yang berbeda.
- **Operator Overloading:** Operator overloading adalah kemampuan untuk mendefinisikan kembali perilaku operator tertentu untuk jenis data kustom. Dalam beberapa bahasa pemrograman seperti C++, operator dapat didefinisikan ulang untuk kelas-kelas yang kita buat sendiri. Namun, Java tidak mendukung operator overloading secara langsung seperti dalam C++. Operator hanya dapat digunakan untuk tipe data bawaan atau objek kelas yang sesuai.

Melewatkan Argumen/Parameter ke Method

Ketika memanggil sebuah metode, kita dapat melewatkannya argumen atau parameter. Dalam Java, semua parameter metode dioperasikan secara "pass by value". Namun, ketika parameter adalah objek, yang dilewatkan adalah referensi ke objek tersebut, sehingga perubahan yang dilakukan dalam objek akan terlihat di luar metode.

. Dalam Java, argumen dapat dilewatkan ke metode secara dua cara:

- **Pass by Value:** Saat melewati argumen secara nilai (pass by value), salinan nilai dari argumen yang asli dikirim ke dalam metode. Ini berarti perubahan yang dibuat dalam parameter metode tidak akan mempengaruhi nilai asli dari argumen.
- **Pass by Reference:** Saat melewati argumen secara referensi (pass by reference), referensi ke objek yang asli dikirim ke dalam metode. Ini berarti perubahan yang dibuat dalam parameter metode akan mempengaruhi nilai asli dari objek tersebut.

Responsibility Driven Design

Responsibility Driven Design (RDD) adalah pendekatan desain yang berfokus pada tanggung jawab (responsibility) yang dimiliki oleh kelas-kelas dalam sistem. Pendekatan ini meminta pengembang untuk memikirkan kelas-kelas dalam sistem dari sudut pandang tanggung jawab yang mereka miliki, dan kemudian mendesain dan mengimplementasikan kelas-kelas tersebut sesuai dengan tanggung jawab mereka. RDD membantu dalam mengorganisir dan mengelompokkan kelas-kelas dalam sistem dengan lebih baik, sehingga memudahkan pemeliharaan dan pengembangan selanjutnya. Ini membantu dalam menerapkan prinsip-prinsip desain yang baik seperti single responsibility principle (SRP) dan high cohesion-low coupling.

II. GUIDED

Guided 1 DEMO.cpp

```
//ramadhan wijaya
//22111102208
#include <iostream>

using namespace std;

class Kelereng {
private:
    int merah;
    int kuning;
public:
    Kelereng(int jum_merah, int jum_kuning);
    void tampil();
    // OverLoading operator
    Kelereng operator+(Kelereng b);
    Kelereng operator-(Kelereng b);
    Kelereng operator*(Kelereng b);
    Kelereng operator/(Kelereng b);
    Kelereng operator%(Kelereng b);
    void operator++(); // Prefix increment
    void operator--(); // Prefix decrement
};

Kelereng::Kelereng(int jum_merah, int jum_kuning) {
    merah = jum_merah;
    kuning = jum_kuning;
}

void Kelereng::tampil() {
    cout << "Jumlah merah = " << merah << ", kuning = " <<
kuning << endl;
}

Kelereng Kelereng::operator+(Kelereng b) {
    Kelereng temp(0, 0);
    temp.merah = merah + b.merah;
    temp.kuning = kuning + b.kuning;
    return temp;
}

Kelereng Kelereng::operator-(Kelereng b) {
```

```

        Kelereng temp(0, 0);
        temp.merah = merah - b.merah;
        temp.kuning = kuning - b.kuning;
        return temp;
    }

Kelereng Kelereng::operator*(Kelereng b) {
    Kelereng temp(0, 0);
    temp.merah = merah * b.merah;
    temp.kuning = kuning * b.kuning;
    return temp;
}

Kelereng Kelereng::operator/(Kelereng b) {
    Kelereng temp(0, 0);
    temp.merah = merah / b.merah;
    temp.kuning = kuning / b.kuning;
    return temp;
}

Kelereng Kelereng::operator%(Kelereng b) {
    Kelereng temp(0, 0);
    temp.merah = merah % b.merah;
    temp.kuning = kuning % b.kuning;
    return temp;
}

void Kelereng::operator++() {
    ++merah;
    ++kuning;
}

void Kelereng::operator--() {
    --merah;
    --kuning;
}

int main() {
    Kelereng Kelereng1(20, 8);
    Kelereng Kelereng2(7, 3);
    Kelereng Kelereng3(0, 0);

    cout << "Objek Kelereng 1" << endl;
    Kelereng1.tampil();
    cout << endl << "Objek Kelereng 2" << endl;
}

```

```

    Kelereng2.tampil();

    // Contoh operator overloading terhadap +
    Kelereng3 = Kelereng1 + Kelereng2;
    cout << endl << "Hasil penjumlahan Kelereng 1 dan
Kelereng 2" << endl;
    Kelereng3.tampil();

    // Contoh operator overloading terhadap -
    Kelereng3 = Kelereng1 - Kelereng2;
    cout << endl << "Hasil pengurangan Kelereng 1 dan
Kelereng 2" << endl;
    Kelereng3.tampil();

    // Contoh operator overloading terhadap *
    Kelereng3 = Kelereng1 * Kelereng2;
    cout << endl << "Hasil perkalian Kelereng 1 dan Kelereng
2" << endl;
    Kelereng3.tampil();

    // Contoh operator overloading terhadap /
    Kelereng3 = Kelereng1 / Kelereng2;
    cout << endl << "Hasil pembagian Kelereng 1 dan Kelereng
2" << endl;
    Kelereng3.tampil();

    // Contoh operator overloading terhadap %
    Kelereng3 = Kelereng1 % Kelereng2;
    cout << endl << "Hasil sisa pembagian Kelereng 1 dan
Kelereng 2" << endl;
    Kelereng3.tampil();

    // Contoh operator overloading terhadap ++
    ++Kelereng1;
    cout << endl << "Hasil ++Kelereng1" << endl;
    Kelereng1.tampil();

    // Contoh operator overloading terhadap --
    --Kelereng2;
    cout << endl << "Hasil --Kelereng2" << endl;
    Kelereng2.tampil();

    cin.get(); // Menunggu input sebelum keluar
    return 0;
}

```



Output :

```
PS D:\SEMESTER_3\Praktikum_PBO\MODUL3> cd "d:\SEMESTER_3\Praktikum_PBO\MODUL3\" ; if ($?) { g++ DEMO1.cpp -o DEMO1 } ; if ($?) { .\DEMO1 }
Objek Kelereng 1
Jumlah merah = 20, kuning = 8

Objek Kelereng 2
Jumlah merah = 7, kuning = 3

Hasil penjumlahan Kelereng 1 dan Kelereng 2
Jumlah merah = 27, kuning = 11

Hasil pengurangan Kelereng 1 dan Kelereng 2
Jumlah merah = 13, kuning = 5

Hasil perkalian Kelereng 1 dan Kelereng 2
Jumlah merah = 140, kuning = 24

Hasil pembagian Kelereng 1 dan Kelereng 2
Jumlah merah = 2, kuning = 2

Hasil sisa pembagian Kelereng 1 dan Kelereng 2
Jumlah merah = 6, kuning = 2

Hasil ++Kelereng1
Jumlah merah = 21, kuning = 9

Hasil --Kelereng2
Jumlah merah = 6, kuning = 2
```

Keterangan :

Menjelaskan fungsi dari operator overloading pada C++, kenapa tidak dalam program java ? karena operator overloading tidak didukung dalam bahasa pemograman java. Fungsi dari setiap methode ini mengambil satu parameter, yaitu objek Kelereng lainnya yang akan dikurangkan/ditambahkan/dimoduluskan/dibagi/dikali/dipangkat dengan nilai variable yang diisi.

Guided 2 bilangan.java

```
/**
 *
```



```

* @author ramadahn wijaya
* @2211102208
* @IF-10-M
*/
class Bilangan {

    private int a;
    private int b;

    public Bilangan(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public void tampil() {
        System.out.println("Nilai bil.a : " + a);
        System.out.println("Nilai bil.b : " + b);
    }
}

//passed by value dengan parameter tipe data primitif

    public void operasi_pass_by_value(int x, int y) {
        x = x * 10;
        y = y + 15;
    }

//passed by reference dengan parameter tipe data class

    public void operasi_pass_by_reference(Bilangan bil) {
        bil.a = bil.a * 10;
        bil.b = bil.b + 15;
    }

    public static void main(String[] args) {
        int x, y;
        Bilangan bil = new Bilangan(10, 20);
//inisialisasi x dan y
        x = 15;
        y = 30;
        System.out.println("Nilai x dan y sebelum passed by value");
        System.out.println("Nilai x : " + x);
        System.out.println("Nilai y : " + y);
        bil.operasi_pass_by_value(x, y); //passed by value

        System.out.println("Nilai x dan y setelah passed by value");
        System.out.println("Nilai x : " + x);
        System.out.println("Nilai y : " + y);
        System.out.println(

```

```

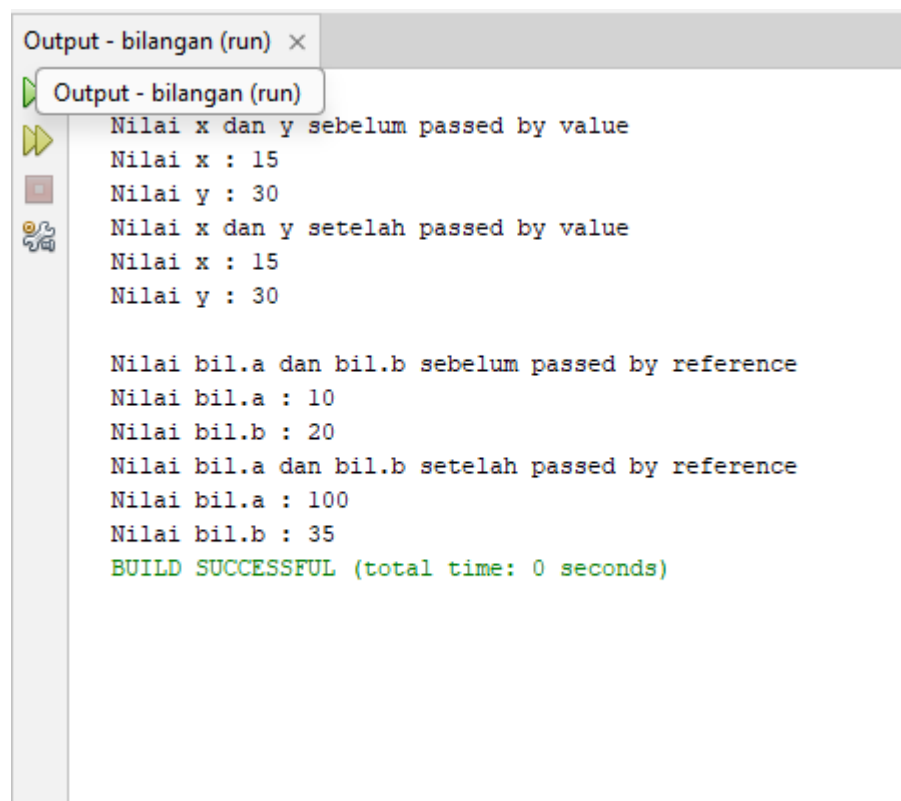
        "\nNilai bil.a dan bil.b sebelum passed by reference ");

bil.tampil();
    bil.operasi_pass_by_reference(bil); //passed by reference
    System.out.println(
        "Nilai bil.a dan bil.b setelah passed by reference ");

bil.tampil();
    }
}

```

Output :



```

Output - bilangan (run) x
Output - bilangan (run)
> Nilai x dan y sebelum passed by value
> Nilai x : 15
> Nilai y : 30
> Nilai x dan y setelah passed by value
> Nilai x : 15
> Nilai y : 30

> Nilai bil.a dan bil.b sebelum passed by reference
> Nilai bil.a : 10
> Nilai bil.b : 20
> Nilai bil.a dan bil.b setelah passed by reference
> Nilai bil.a : 100
> Nilai bil.b : 35
BUILD SUCCESSFUL (total time: 0 seconds)

```

Keterangan :

Program diatas adalah program sederhana menunjukan fungsi passed by value dan passed by reference secara bersamaan seperti yang dijelaskan di dasar teori operasi pass_by_val() berfungsi agar variable asli tidak diubah oleh operasi dalam metode. Jadi menjaga kekonsistenan nilai dari luar metode tersebut. Contohnya bil.a dan bil.b tidak berubah setelah dilakukan operasi tersebut. Berbeda dengan operasi pass_by_reference(); Fungsi "pass by reference" adalah mekanisme di mana alamat memori atau referensi dari sebuah variabel (bukan nilainya) yang dilewatkan ke sebuah fungsi

atau metode. Dengan menggunakan "pass by reference", perubahan yang dilakukan pada parameter di dalam fungsi akan mempengaruhi nilai asli variabel di luar fungsi tersebut.

Guided 3 DEMO2.cpp

```
//ramadhan wijaya
//22111102208
// Prinsip-Prinsip Perancangan Kelas
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <string.h>

using namespace std;
/***** Mendeklarasikan Kelas *****/
class Buku
{
private: // encapsulation atributes dari kelas
    char pengarang[24];
    char judul[20];
    int jmlh_halaman;
    float diskon;
    double Harga;

public: // visibilitas methods dari kelas
    Buku(); // constructor
    ~Buku(); // destructor

    // fungsi-fungsi mutator
    void setPengarang(char *Pengarang);
    void setJudul(char *Judul);
    void setJmlhHalaman(int JmlhHalaman);
    // fungsi-fungsi accessor
    char *getPengarang();
    char *getJudul();
    int getJmlhHalaman();
    // method overloading
    void setInfo(float dskn, double HargaBuku);
```

```

        void setInfo(double HargaBuku);
        void cetak();
    };
    /****** Mendeklarasikan Methods dari Kelas *****/
    Buku::Buku() // constructor
    {
        cout << "Konstruktor buku dijalankan...." << endl;
    }

    Buku::~~Buku() // destructor
    {
        // perintah di bawah ini hanya menunjukkan bahwa
destruktor dipanggil di akhir program
        cout<< "Destruktor buku dijalankan...." << endl;
        getch();
    }

    // fungsi-fungsi mutator
    void Buku::setPengarang(char *Pengarang)
    {
        strcpy(pengarang, Pengarang);
    }
    void Buku::setJudul(char *Judul)
    {
        strcpy(judul, Judul);
    }
    void Buku::setJmlhHalaman(int JmlhHalaman)
    {
        jmlh_halaman = JmlhHalaman;
    }

    // fungsi-fungsi accessor
    char *Buku::getPengarang()
    {
        return pengarang;
    }
    char *Buku::getJudul()
    {
        return judul;
    }
    int Buku::getJmlhHalaman()
    {
        return jmlh_halaman;
    }
    // method overloading

```

```

void Buku::setInfo(float dskn, double HargaBuku)
{
    diskon = dskn;
    Harga = HargaBuku - (diskon * HargaBuku);
}

void Buku::setInfo(double HargaBuku)
{
    diskon = 0.1;
    Harga = HargaBuku - (diskon * HargaBuku);
}

void Buku::cetak()
{
    cout << setiosflags(ios::fixed); // manipulasi digit
    pecahan
    cout << "\nJudul Buku : " << getJudul() << endl;
    cout << "Pengarang : " << getPengarang() << endl;
    cout << "Jumlah halaman Buku : " << getJmlhHalaman() <<
    " halaman "<<endl;
    cout << "Diskon Buku dibeli : " << setprecision(2) <<
    diskon << endl;
    cout << "Harga Buku : "<< setprecision(2)<<Harga<<endl;
}

/***** Main Program *****/
int main(){
    Buku novel, fiksi, folklore, novel2;
    cout << endl;
    novel.setJudul("Interstellar");
    novel.setPengarang("cristopher nolan");
    novel.setInfo(0.2, 45000);
    novel.setJmlhHalaman(123);
    novel.cetak();

    fiksi.setJudul("vent");
    fiksi.setPengarang("caroline mcquen");
    fiksi.setJmlhHalaman(202);
    fiksi.setInfo(79000);
    fiksi.cetak();

    folklore.setJudul("diao chan");
    folklore.setPengarang("somebody");
    folklore.setJmlhHalaman(142);
    folklore.setInfo(120000);
}

```

```

        folklore.cetak();

        novel12.setJudul("perang 3 dinasti");
        novel12.setPengarang("dr.hermawan");
        novel12.setJmlhHalaman(603);
        novel12.setInfo(22500000);
        novel12.cetak();

        getch();
        return 0;
    }

```

Output :

```

Konstruktor buku dijalankan...
Konstruktor buku dijalankan...
Konstruktor buku dijalankan...
Konstruktor buku dijalankan...

Judul Buku : Interstellar
Pengarang : cristopher nolan
Jumlah halaman Buku : 123 halaman
Diskon Buku dibeli : 0.20
Harga Buku : 36000.00

Judul Buku : vent
Pengarang : caroline mcquen
Jumlah halaman Buku : 202 halaman
Diskon Buku dibeli : 0.10
Harga Buku : 71100.00

Judul Buku : diao chan
Pengarang : somebody
Jumlah halaman Buku : 142 halaman
Diskon Buku dibeli : 0.10
Harga Buku : 108000.00

Judul Buku : perang 3 dinasti
Pengarang : dr.hermawan
Jumlah halaman Buku : 603 halaman
Diskon Buku dibeli : 0.10
Harga Buku : 20249999.97

```

Keterangan :

Program diatas menjelaskan bagaimana cara kerja mutator dan accessor. Pada pemanggilan fungsi accessor dan mutator diperlukan :: untuk Mengakses Anggota Kelas: Ketika digunakan dengan nama kelas, :: digunakan untuk mengakses anggota kelas yang bersifat statis (static members) atau anggota kelas dari luar kelas itu sendiri, Definisi Fungsi Anggota Kelas di Luar Kelas: Operator :: juga digunakan untuk mendefinisikan fungsi anggota kelas di luar definisi kelas itu sendiri, Mengakses Ruang Lingkup Global: Operator :: digunakan untuk mengakses variabel atau fungsi global dari dalam fungsi atau kelas, dan terakhir namespace Resolution: Operator :: juga digunakan untuk mengakses elemen-elemen dari namespace tertentu.

Pada bagian int main() berfungsi untuk mendeklarasikan menginisialisasikan program yang membuat object buku secara statik. dan sesuai dengan fungsi fungsi yang ditentukan.

Guided 4 buku.java

```
/**
 * @author ramadhan wijaya
 * @2211102208
 * @IF-10-M
 */
//Prinsip-Prinsip Perancangan Kelas /* Mendeklarasikan Kelas */
import java.io.BufferedReader;
import java.io.InputStreamReader;

class Buku {
    private String pengarang;
    private String judul;
    private int jmlh_Halaman;
    private float diskon;
    private double harga;

    public Buku() { // constructor
        System.out.println("Konstruktor buku dijalankan...");
    }

    public void setPengarang(String Pengarang) {
        pengarang = Pengarang;
    }

    public void setJudul(String Judul) {
```

```

        judul = Judul;
    }

    public void setJmlhHalaman(int JmlhHalaman) {
        jmlh_Halaman = JmlhHalaman;
    }

    public String getPengarang() {
        return pengarang;
    }

    public String getJudul() {
        return judul;
    }

    public int getJmlhHalaman() {
        return jmlh_Halaman;
    }

    public void setInfo(float dskn, double HargaBuku) {
        diskon = dskn;
        harga = HargaBuku - (HargaBuku * diskon);
    }

    public void setInfo(double HargaBuku) {
        diskon = 0.1F;
        harga = HargaBuku - (HargaBuku * diskon);
    }

    public void cetak() {
        System.out.println("Judul Buku : " + getJudul());
        System.out.println("Pengarang Buku : " + getPengarang());
        System.out.println("Jumlah halaman Buku : " + getJmlhHalaman()
+ " halaman");
        System.out.println("Diskon Buku : " + diskon);
        System.out.println("Harga Buku : " + harga);
        System.out.println();
    }

    public static void main(String[] args) {
        Buku novel, fiksi;
        novel = new Buku();
        fiksi = new Buku();
        System.out.println();
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

```

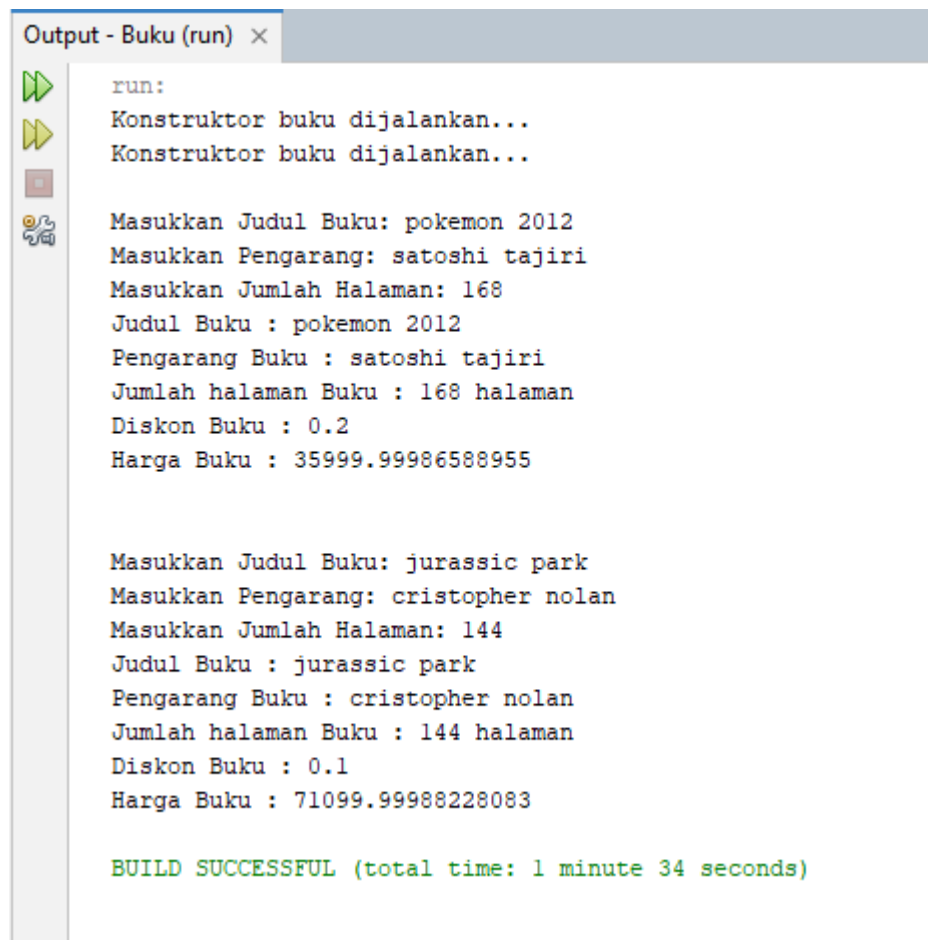


```

try {
    System.out.print("Masukkan Judul Buku: ");
    novel.setJudul(br.readLine().toString());
    System.out.print("Masukkan Pengarang: ");
    novel.setPengarang(br.readLine().toString());
    novel.setInfo(0.2f, 45000);
    System.out.print("Masukkan Jumlah Halaman: ");
    novel.setJmlhHalaman(Integer.parseInt(br.readLine()));
    novel.cetak();
    System.out.println();
    System.out.print("Masukkan Judul Buku: ");
    fiksi.setJudul(br.readLine().toString());
    System.out.print("Masukkan Pengarang: ");
    fiksi.setPengarang(br.readLine().toString());
    fiksi.setInfo(79000);
    System.out.print("Masukkan Jumlah Halaman: ");
    fiksi.setJmlhHalaman(Integer.parseInt(br.readLine()));
    fiksi.cetak();
} catch (Exception ex) {
    System.out.println("Error: " + ex.getMessage());
}
}
}

```

Output :



```
run:
Konstruktor buku dijalankan...
Konstruktor buku dijalankan...

Masukkan Judul Buku: pokemon 2012
Masukkan Pengarang: satoshi tajiri
Masukkan Jumlah Halaman: 168
Judul Buku : pokemon 2012
Pengarang Buku : satoshi tajiri
Jumlah halaman Buku : 168 halaman
Diskon Buku : 0.2
Harga Buku : 35999.99986588955

Masukkan Judul Buku: jurassic park
Masukkan Pengarang: cristopher nolan
Masukkan Jumlah Halaman: 144
Judul Buku : jurassic park
Pengarang Buku : cristopher nolan
Jumlah halaman Buku : 144 halaman
Diskon Buku : 0.1
Harga Buku : 71099.99988228083

BUILD SUCCESSFUL (total time: 1 minute 34 seconds)
```

Keterangan :

Program diatas adalah program membuat sebuah keterangan pada(mungkin) sebuah toko buku yang mengadakan diskon Dimana diskonnya adalah 20% untuk buku pertama dan 10% untuk buku ke dua yang dimana harganya sudah ditentukan melalui setInfo(), otomatis harganya di terpotong sekian persen dari harga yang kita tentukan. Pada program terlihat jelas ada konstruktor buku() dan banyak fungsi dan prosedur yaitu public void setPengarangan(),setJudul(),setJmlhHalaman(),getJmlHlaman(),setInfo(), dan terakhir cetak().

Yang menarik disini adalah br.readLine() yang digunakan untuk membaca teks dari input stream (dalam hal ini, input dari konsol). Ketika br.readLine() dieksekusi, program akan menunggu hingga pengguna mengetikkan sesuatu di konsol dan menekan tombol "Enter". Setelah itu, teks yang dimasukkan oleh pengguna akan dibaca dan dikembalikan sebagai string.

Ada juga `Integer.parseInt()`. Fungsi `br.readLine()` mengembalikan sebuah string yang dibaca dari input stream. Namun, jika Anda ingin menggunakan nilai tersebut sebagai integer (angka bulat), Anda perlu mengonversinya dari tipe data string menjadi tipe data integer. Itulah fungsi dari `Integer.parseInt()`. Jadi ketika di eksekusi, `Integer.parseInt(br.readLine())` membaca sebuah baris teks dari input stream yang diberikan oleh `BufferedReader (br)`, kemudian mengonversi nilai teks tersebut menjadi tipe data integer. Hasil dari operasi tersebut adalah nilai integer yang dapat digunakan dalam program. Dalam konteks ini, nilai tersebut kemudian digunakan untuk mengatur jumlah halaman dari objek fiksi menggunakan metode `setJmlhHalaman()`

III. UNGUIDED

Buatlah program dalam bahasa **Java** dan **C++** untuk mencatat data pegawai. Atribut data pegawai meliputi: **NIP, nama, alamat, jumlah hari lembur, gaji pokok dan total gaji.**

Nilai-nilai dari atribut pegawai diset dalam program. Lakukan perhitungan total gaji lalu tampilkan data pegawai tersebut.

Ketentuan:

- Beri comment header pad program anda : nim, nama, kelas **(bobot 10 %)**
- Ada konstruktor dan destructor **(bobot 30 %)**
- Ada fungsi accesor dan fungsi mutator **(bobot 40 %)**
- Method overloading terhadap perhitungan total gaji pegawai. **(bobot 20 %)**
Total gaji = gaji pokok + (gaji pokok * jumlah hari lembur * 0.01)

Source code :

```
/**
 *
 * @author ramadhan wijaya
 * @2211102208
 * @IF-10-M
 */
public class Pegawai {
    // Atribut data pegawai
    private String NIP;
    private String nama;
    private String alamat;
    private int jumlahHariLembur;
    private double gajiPokok;
    private double totalGaji;

    // Constructor
    public Pegawai(String NIP, String nama, String alamat, int
jumlahHariLembur, double gajiPokok) {
        this.NIP = NIP;
        this.nama = nama;
        this.alamat = alamat;
        this.jumlahHariLembur = jumlahHariLembur;
        this.gajiPokok = gajiPokok;
    }

    // Destructor
    protected void finalize() throws Throwable {
```

```

        System.out.println("Objek Pegawai dengan NIP " + this.NIP + "
dihapus.");
        super.finalize();
    }

    // Fungsi accessor
    public String getNIP() {
        return NIP;
    }

    public String getNama() {
        return nama;
    }

    public String getAlamat() {
        return alamat;
    }

    public int getJumlahHariLembur() {
        return jumlahHariLembur;
    }

    public double getGajiPokok() {
        return gajiPokok;
    }

    public double getTotalGaji() {
        return totalGaji;
    }

    // Fungsi mutator
    public void setJumlahHariLembur(int jumlahHariLembur) {
        this.jumlahHariLembur = jumlahHariLembur;
    }

    public void setGajiPokok(double gajiPokok) {
        this.gajiPokok = gajiPokok;
    }

    // Method overloading untuk perhitungan total gaji
    public void hitungTotalGaji() {
        totalGaji = gajiPokok + (gajiPokok * jumlahHariLembur * 0.01);
    }

    public void hitungTotalGaji(double lemburRate) {
        totalGaji = gajiPokok + (gajiPokok * jumlahHariLembur * lemburRate);
    }

```

```

    }

    // Method untuk menampilkan data pegawai
    public void tampilkanData() {
        System.out.println("NIP: " + NIP);
        System.out.println("Nama: " + nama);
        System.out.println("Alamat: " + alamat);
        System.out.println("Jumlah Hari Lembur: " + jumlahHariLembur);
        System.out.println("Gaji Pokok: " + gajiPokok);
        System.out.println("Total Gaji: " + totalGaji);
    }

    public static void main(String[] args) {
        // Membuat objek pegawai
        Pegawai pegawai0 = new Pegawai("123456", "Mc. gregor", "Jl. bagor", 5,
5000000);
        Pegawai pegawai1 = new Pegawai("543210", "Yuiji", "Jl. pangrango
no.42", 3, 3000000);

        // Menghitung total gaji
        pegawai0.hitungTotalGaji();
        pegawai1.hitungTotalGaji();

        // Menampilkan data pegawai
        pegawai0.tampilkanData();
        System.out.print("\n"); // Karakter tabulasi

        pegawai1.tampilkanData();
        System.out.print("\n"); // Karakter tabulasi
    }
}

```

Output :

```
Output - pegawai (run) ×
run:
NIP: 123456
Nama: Mc. gregor
Alamat: Jl. bagor
Jumlah Hari Lembur: 5
Gaji Pokok: 5000000.0
Total Gaji: 5250000.0

NIP: 543210
Nama: Yuiji
Alamat: Jl. pangrango no.42
Jumlah Hari Lembur: 3
Gaji Pokok: 3000000.0
Total Gaji: 3090000.0

BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Program diatas adalah program untuk pencatatan uang gaji tambahan dari lemburnya. Terpampang jelas pada method/fungsi/procedure hitungTotalGaji() terdapat rumus gaji pokok yang ditambahkan dengan gaji lembur yang dinamai dengan lemburRate yang menggunakan overloading sesuai ketentuan soal.